FIG. 1 PRIOR ART

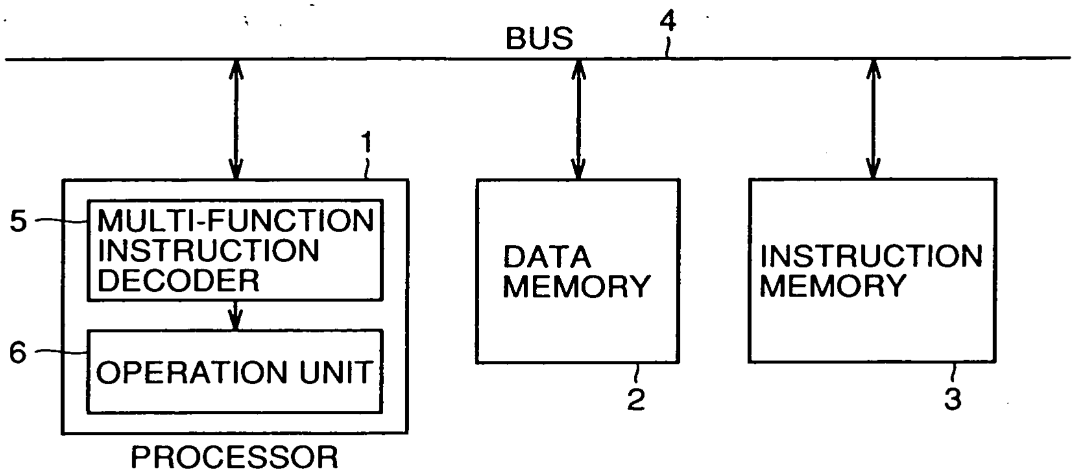


FIG.2

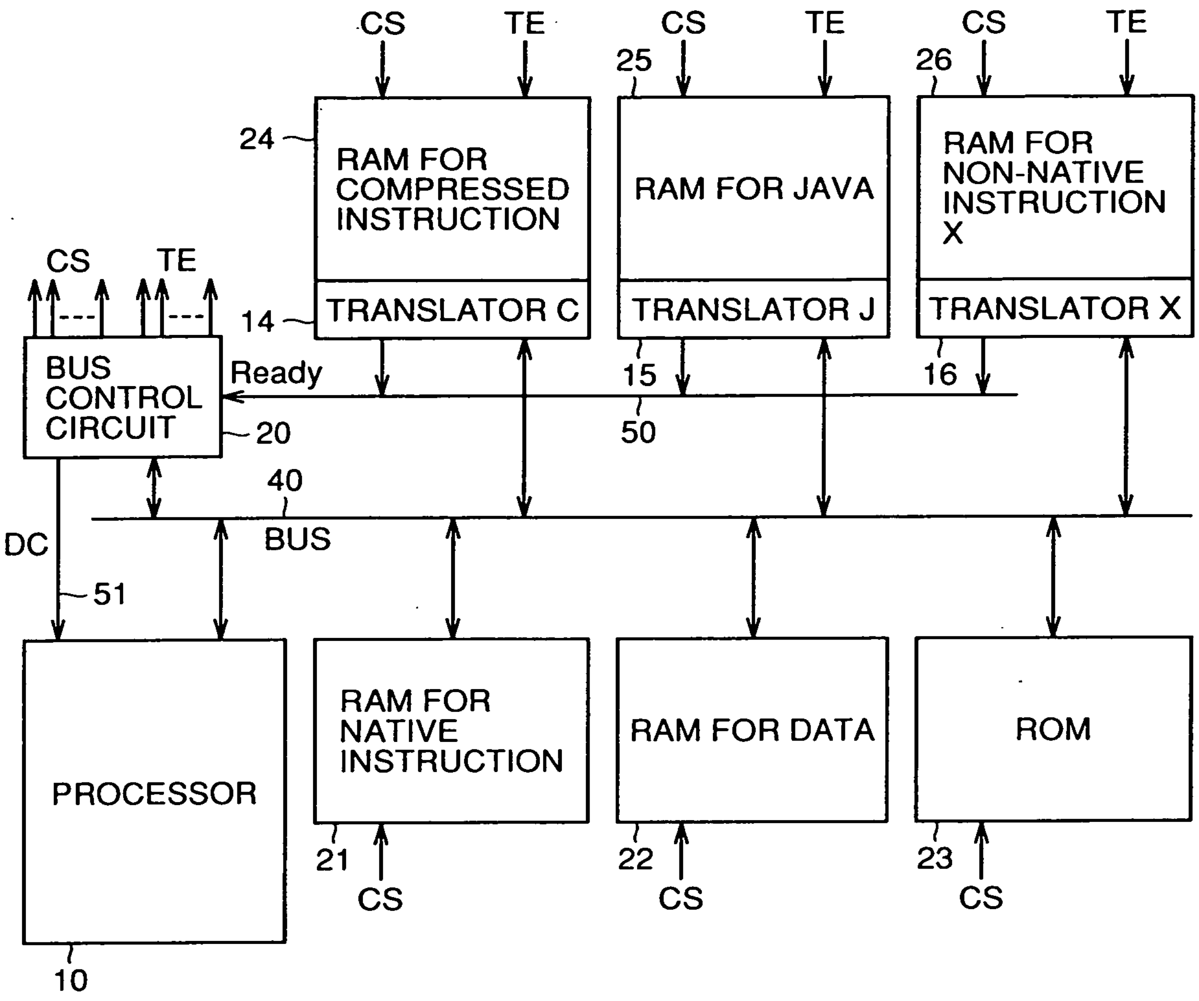

CS
TE
CS
TE
CS
TE
25
26
24
RAM FOR COMPRESSED INSTRUCTION
RAM FOR JAVA
RAM FOR NON-NATIVE INSTRUCTION X
CS
TE
14
TRANSLATOR C
TRANSLATOR J
TRANSLATOR X
BUS CONTROL CIRCUIT
Ready
15
16
20
50
40
BUS
DC
51
PROCESSOR
RAM FOR NATIVE INSTRUCTION
RAM FOR DATA
ROM
21
22
23
CS
CS
CS
10

FIG.3

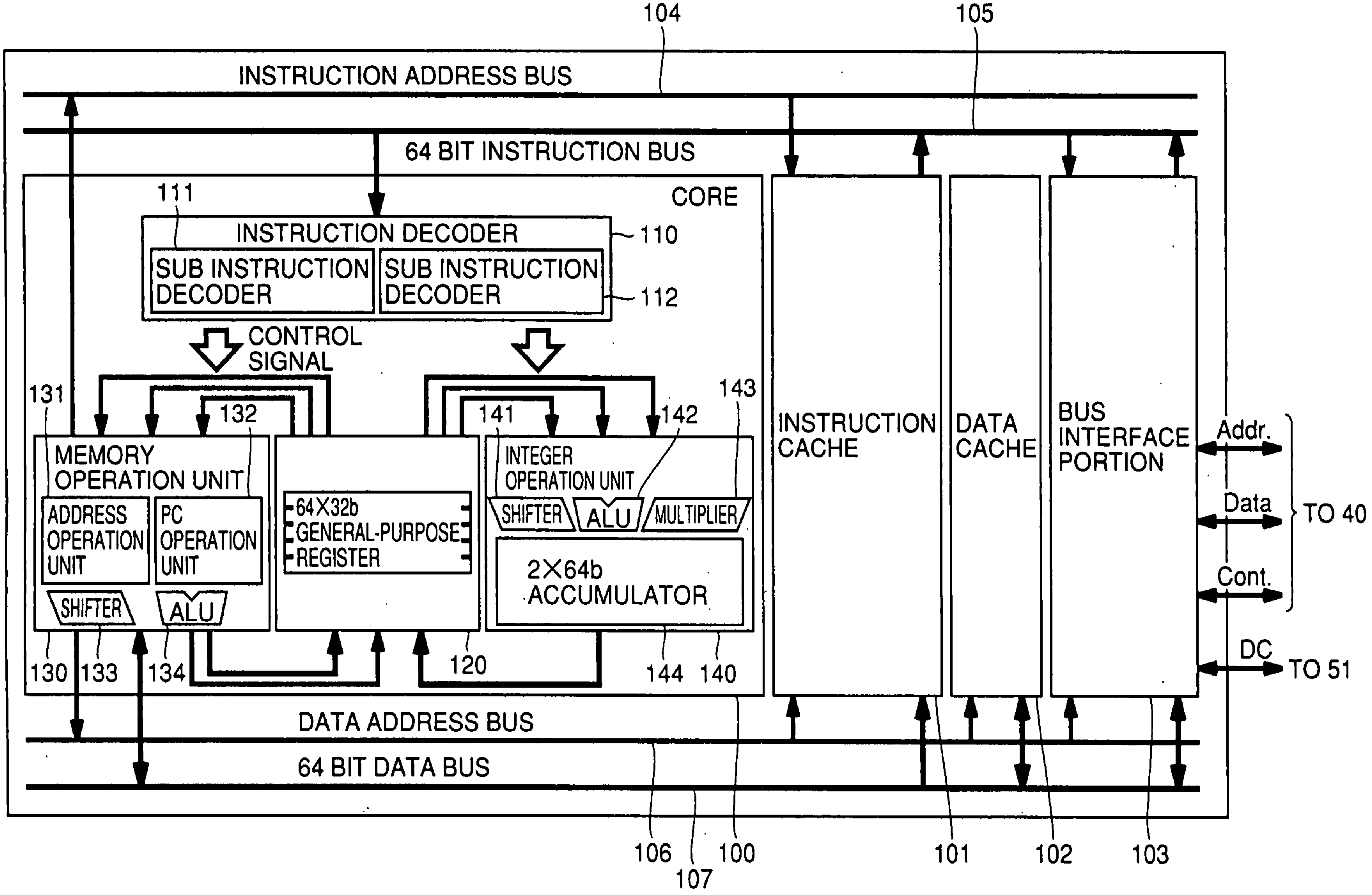

104
105
INSTRUCTION ADDRESS BUS
64 BIT INSTRUCTION BUS
CORE
111
INSTRUCTION DECODER
110
SUB INSTRUCTION DECODER
SUB INSTRUCTION DECODER
112
CONTROL SIGNAL
131
132
141
142
143
MEMORY OPERATION UNIT
ADDRESS OPERATION UNIT
PC OPERATION UNIT
SHIFTER
ALU
64X32b GENERAL-PURPOSE REGISTER
INTEGER OPERATION UNIT
SHIFTER
ALU
MULTIPLIER
2X64b ACCUMULATOR
130
133
134
120
144
140
INSTRUCTION CACHE
DATA CACHE
BUS INTERFACE PORTION
Addr.
Data
Cont.
TO 40
DC
TO 51
DATA ADDRESS BUS
64 BIT DATA BUS
106
107
100
101
102
103

FIG.4

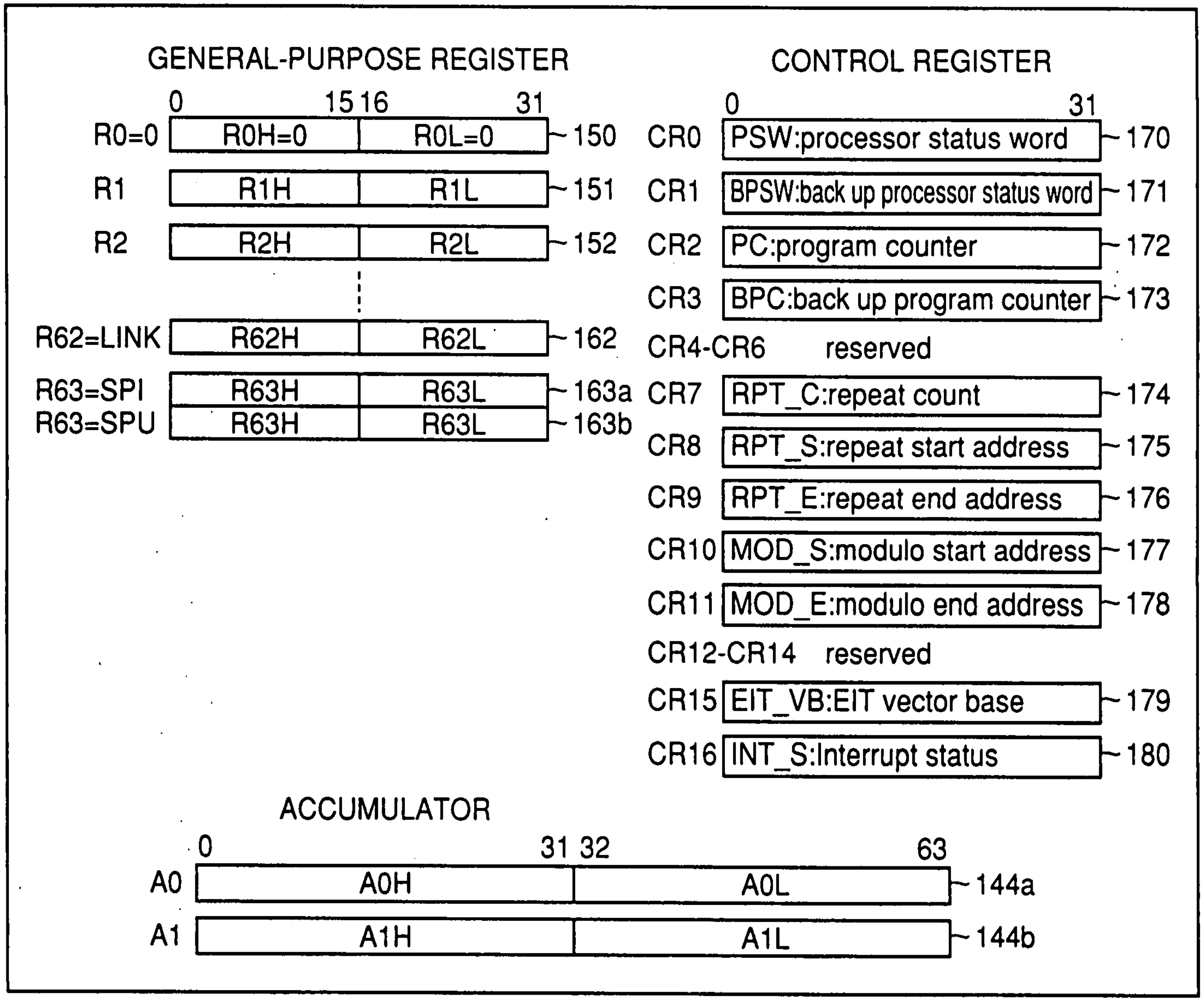

GENERAL-PURPOSE REGISTER
0
15 16
31
R0=0
R0H=0
R0L=0
150
R1
R1H
R1L
151
R2
R2H
R2L
152
R62=LINK
R62H
R62L
162
R63=SPI
R63H
R63L
163a
R63=SPU
R63H
R63L
163b
CONTROL REGISTER
0
31
CR0
PSW:processor status word
170
CR1
BPSW:back up processor status word
171
CR2
PC:program counter
172
CR3
BPC:back up program counter
173
CR4-CR6
reserved
CR7
RPT_C:repeat count
174
CR8
RPT_S:repeat start address
175
CR9
RPT_E:repeat end address
176
CR10
MOD_S:modulo start address
177
CR11
MOD_E:modulo end address
178
CR12-CR14
reserved
CR15
EIT_VB:EIT vector base
179
CR16
INT_S:Interrupt status
180
ACCUMULATOR
0
31 32
63
A0
A0H
A0L
144a
A1
A1H
A1L
144b

FIG.5

0	170a				170b		7	8							15
SM	0	0	0	0	IE	RP	MD	0	0	0	0	0	0	0	0

16	170c		170d				23	24							31
0	F0	0	F1	0	F2	0	F3	0	F4	0	F5	0	F6	0	F7

- ● SM:Stack mode
  SM=0 Interrupt mode.SPI is used.
  SM=1 User mode.SPU is used.
- ● IE:Interrupt enable
  IE=0 Interrupts are masked.
  IE=1 Interrupts are accepted.
- ● RP:Repeat enable
  RP=0 A block repeat is inactive.
  RP=1 A block repeat is active.
- ● MD:Modulo enable
  MD=0 Modulo addressing is disabled.
  MD=1 Modulo addressing is enabled.
- ● F0 :Execution control flag #0
- ● F1 :Execution control flag #1
- ● F2 :General flag
- ● F3 :General flag
- ● F4(S) :Saturation flag
- ● F5(V) :Overflow flag
- ● F6(VA) :Accumulated Overflow flag
  VA is cleared by a reset interrupt and MVTSYS instruction.
- ● F7(C) :Carry/Borrow flag

FIG.6

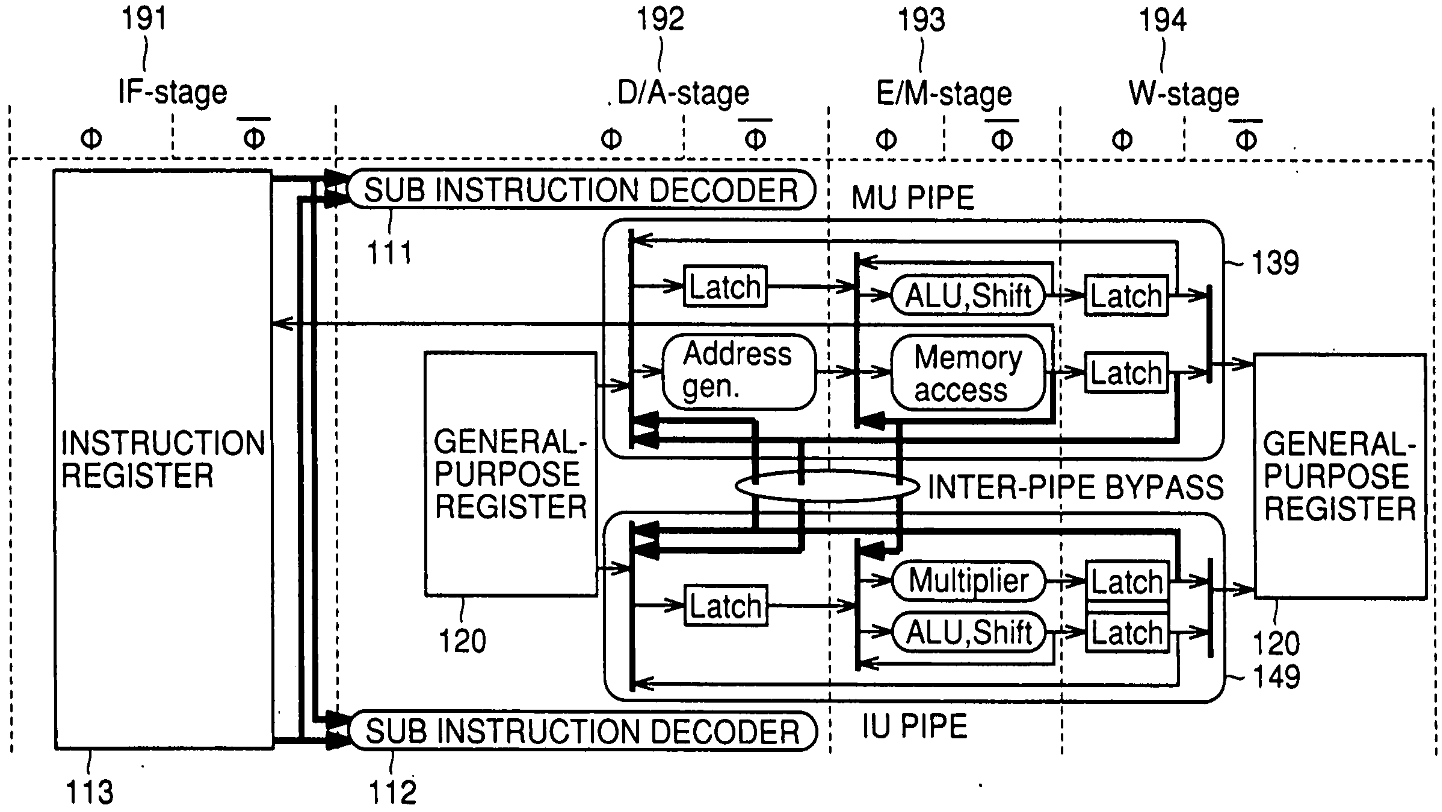
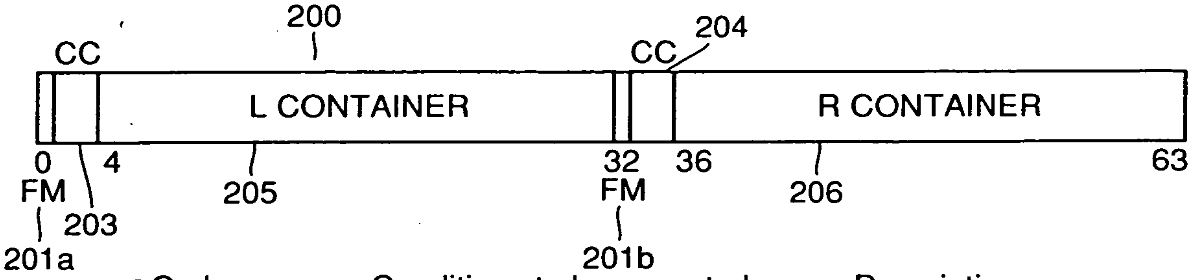

191
IF-stage
Φ
$\overline{Φ}$
192
D/A-stage
Φ
$\overline{Φ}$
193
E/M-stage
Φ
$\overline{Φ}$
194
W-stage
Φ
$\overline{Φ}$
INSTRUCTION REGISTER
113
SUB INSTRUCTION DECODER
111
SUB INSTRUCTION DECODER
112
GENERAL-PURPOSE REGISTER
120
MU PIPE
139
Latch
Address gen.
ALU,Shift
Memory access
Latch
Latch
INTER-PIPE BYPASS
IU PIPE
149
Latch
Multiplier
ALU,Shift
Latch
Latch
GENERAL-PURPOSE REGISTER
120

FIG.7



●Code	Conditions to be executed	Description
CC = 000	Always	---
CC = 001	F0 = T and F1 = don't care	/TX
CC = 010	F0 = F and F1 = don't care	/FX
CC = 011	F0 = don't care and F1 = T	/XT
CC = 100	F0 = don't care and F1 = F	/XF
CC = 101	F0 = T and F1 = T	/TT
CC = 110	F0 = T and F1 = F	/TF
CC = 111	Reserved	---

FM	Number of sub-instructions	Issuing Order	
		L-container	R-container
00	two	1st	1st
01	two	1st	2nd
10	two	2nd	1st
11	one	1st	----

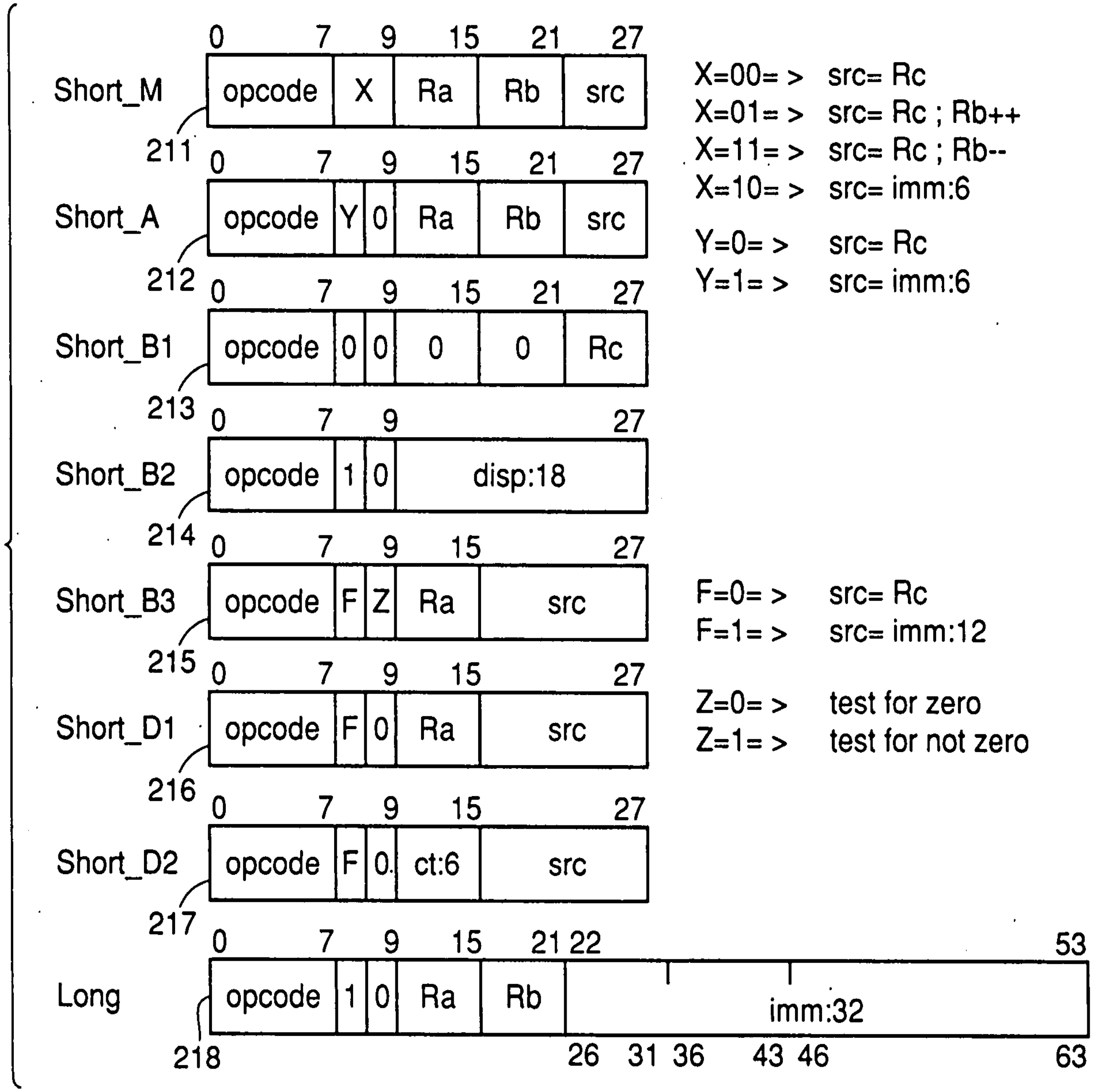


FIG.8

Short_M
211
0 7 9 15 21 27
opcode X Ra Rb src
X=00=> src= Rc
X=01=> src= Rc ; Rb++
X=11=> src= Rc ; Rb--
X=10=> src= imm:6
Short_A
212
0 7 9 15 21 27
opcode Y 0 Ra Rb src
Y=0=> src= Rc
Y=1=> src= imm:6
Short_B1
213
0 7 9 15 21 27
opcode 0 0 0 0 Rc
Short_B2
214
0 7 9 27
opcode 1 0 disp:18
Short_B3
215
0 7 9 15 27
opcode F Z Ra src
F=0=> src= Rc
F=1=> src= imm:12
Short_D1
216
0 7 9 15 27
opcode F 0 Ra src
Z=0=> test for zero
Z=1=> test for not zero
Short_D2
217
0 7 9 15 27
opcode F 0 ct:6 src
Long
218
0 7 9 15 21 22 53
opcode 1 0 Ra Rb imm:32
26 31 36 43 46 63

FIG.9

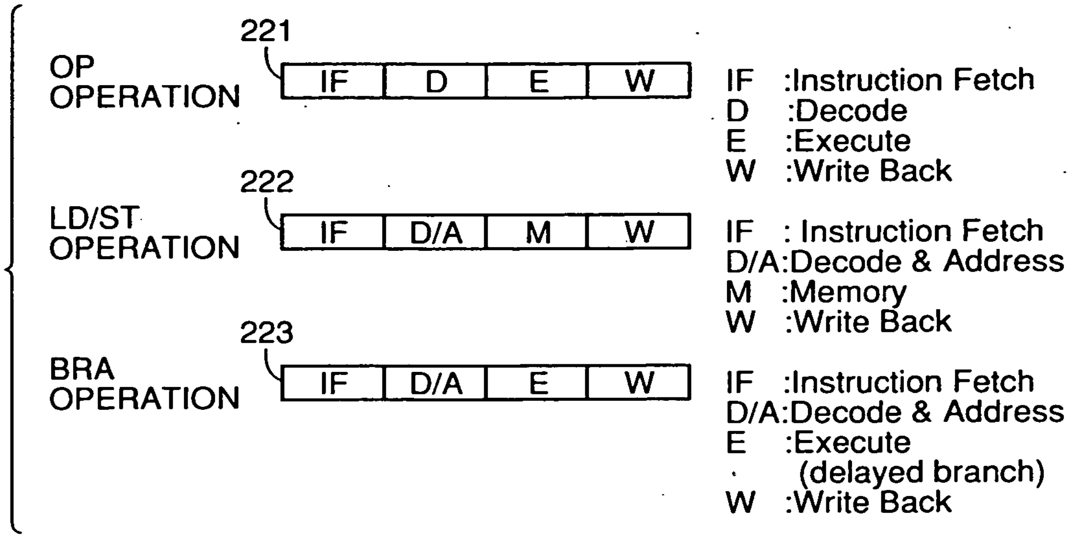

221
OP OPERATION
IF | D | E | W
IF :Instruction Fetch
D :Decode
E :Execute
W :Write Back
222
LD/ST OPERATION
IF | D/A | M | W
IF : Instruction Fetch
D/A:Decode & Address
M :Memory
W :Write Back
223
BRA OPERATION
IF | D/A | E | W
IF :Instruction Fetch
D/A:Decode & Address
E :Execute
(delayed branch)
W :Write Back

FIG.10

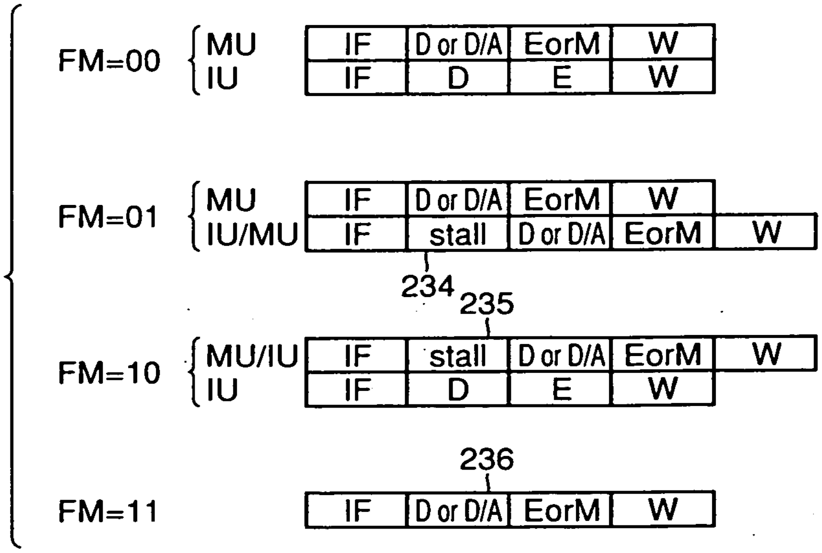
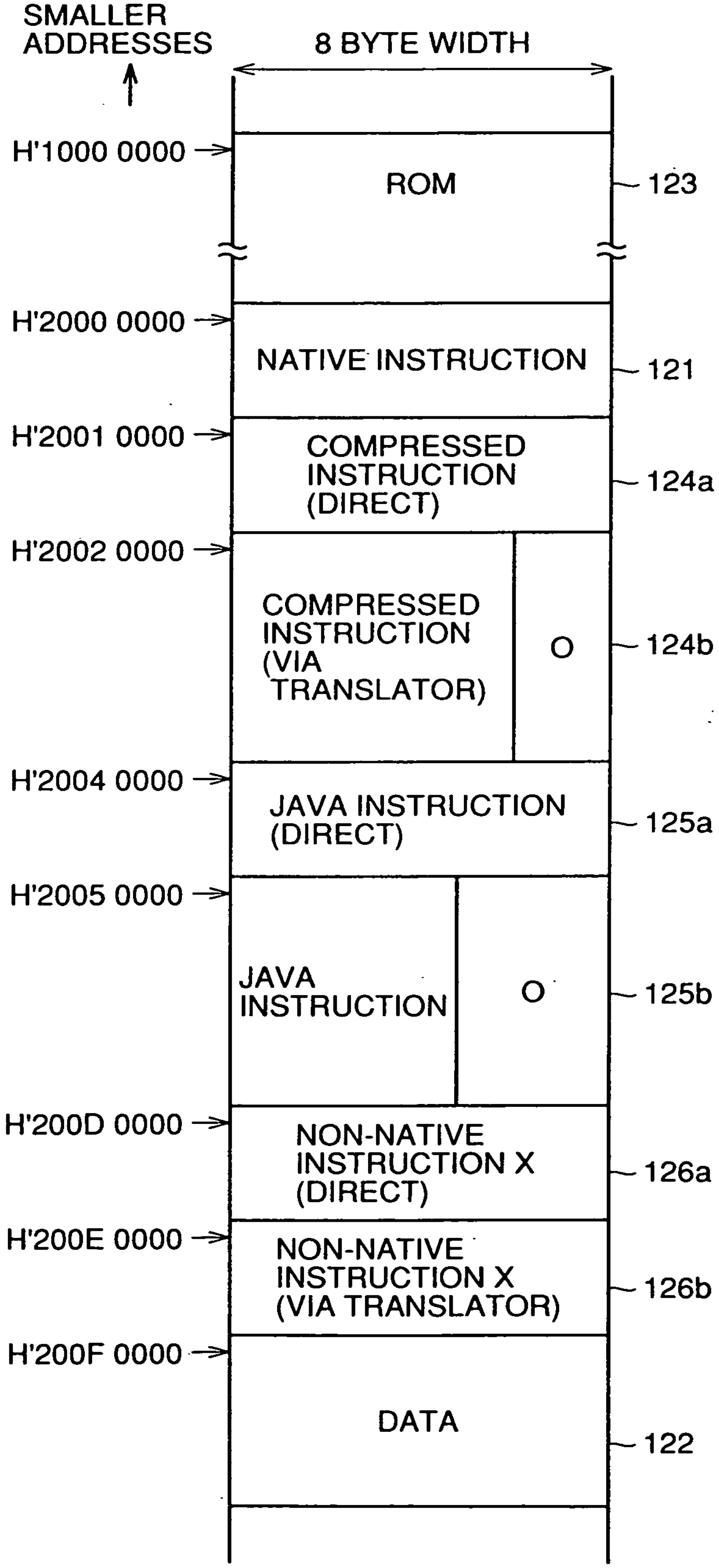

FM=00
MU
IF | D or D/A | EorM | W
IU
IF | D | E | W
FM=01
MU
IF | D or D/A | EorM | W
IU/MU
IF | stall | D or D/A | EorM | W
234
235
FM=10
MU/IU
IF | stall | D or D/A | EorM | W
IU
IF | D | E | W
236
FM=11
IF | D or D/A | EorM | W

FIG.11



SMALLER ADDRESSES
8 BYTE WIDTH
H'1000 0000
ROM
123
H'2000 0000
NATIVE INSTRUCTION
121
H'2001 0000
COMPRESSED INSTRUCTION (DIRECT)
124a
H'2002 0000
COMPRESSED INSTRUCTION (VIA TRANSLATOR)
O
124b
H'2004 0000
JAVA INSTRUCTION (DIRECT)
125a
H'2005 0000
JAVA INSTRUCTION
O
125b
H'200D 0000
NON-NATIVE INSTRUCTION X (DIRECT)
126a
H'200E 0000
NON-NATIVE INSTRUCTION X (VIA TRANSLATOR)
126b
H'200F 0000
DATA
122

FIG. 12

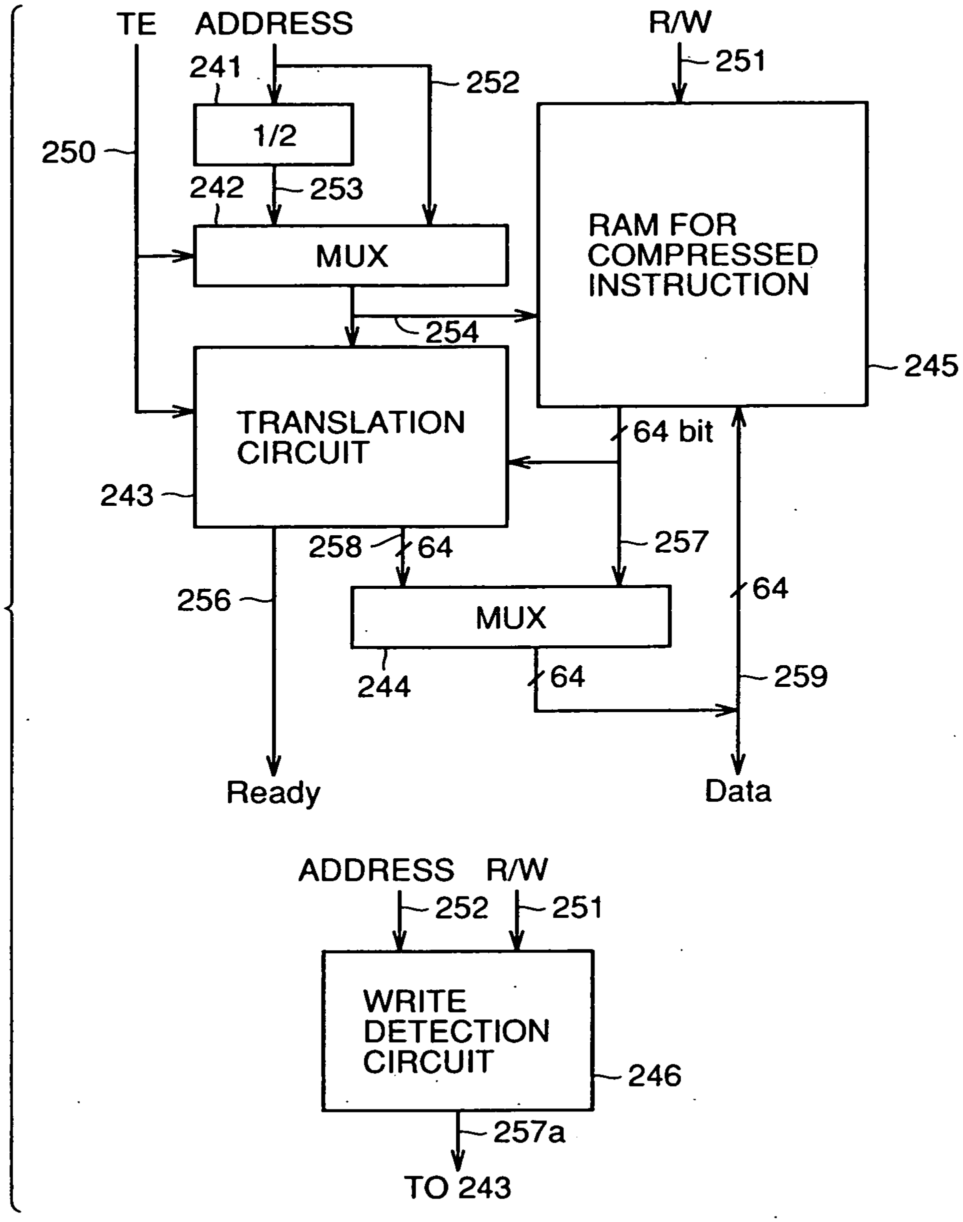
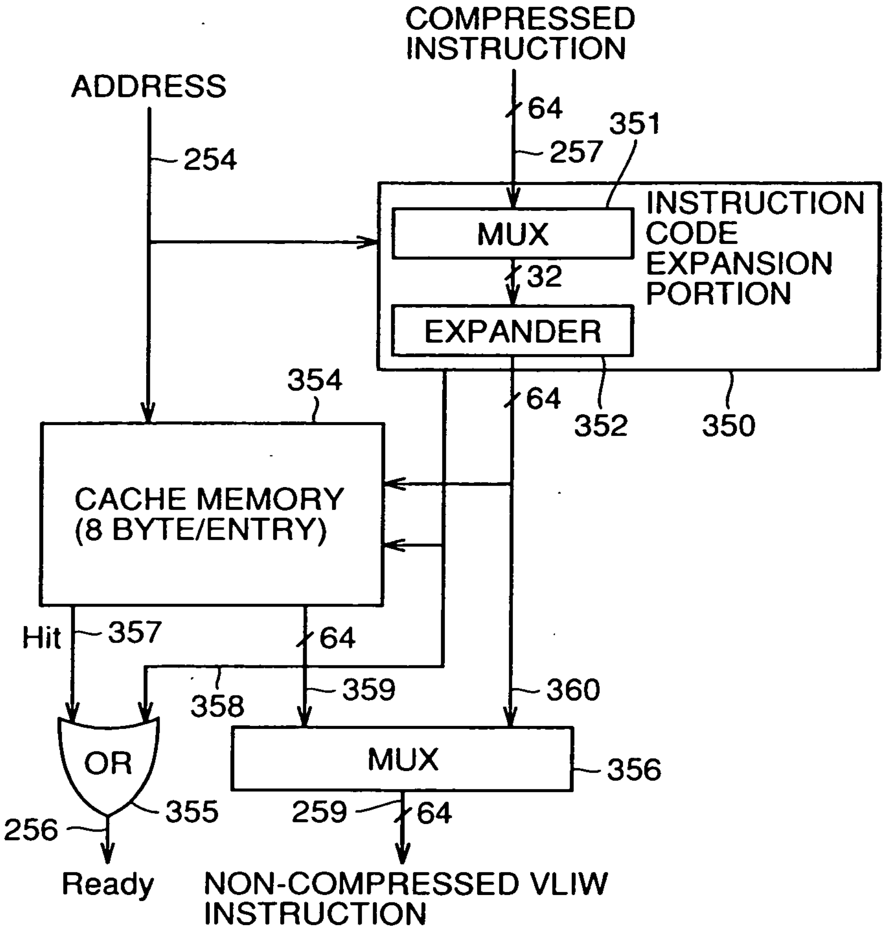

TE
ADDRESS
R/W
241
252
251
1/2
250
242
253
MUX
RAM FOR COMPRESSED INSTRUCTION
254
245
TRANSLATION CIRCUIT
64 bit
243
258
64
257
256
64
MUX
244
64
259
Ready
Data
ADDRESS
R/W
252
251
WRITE DETECTION CIRCUIT
246
257a
TO 243

FIG. 13



TRANSLATION CIRCUIT

FIG.14

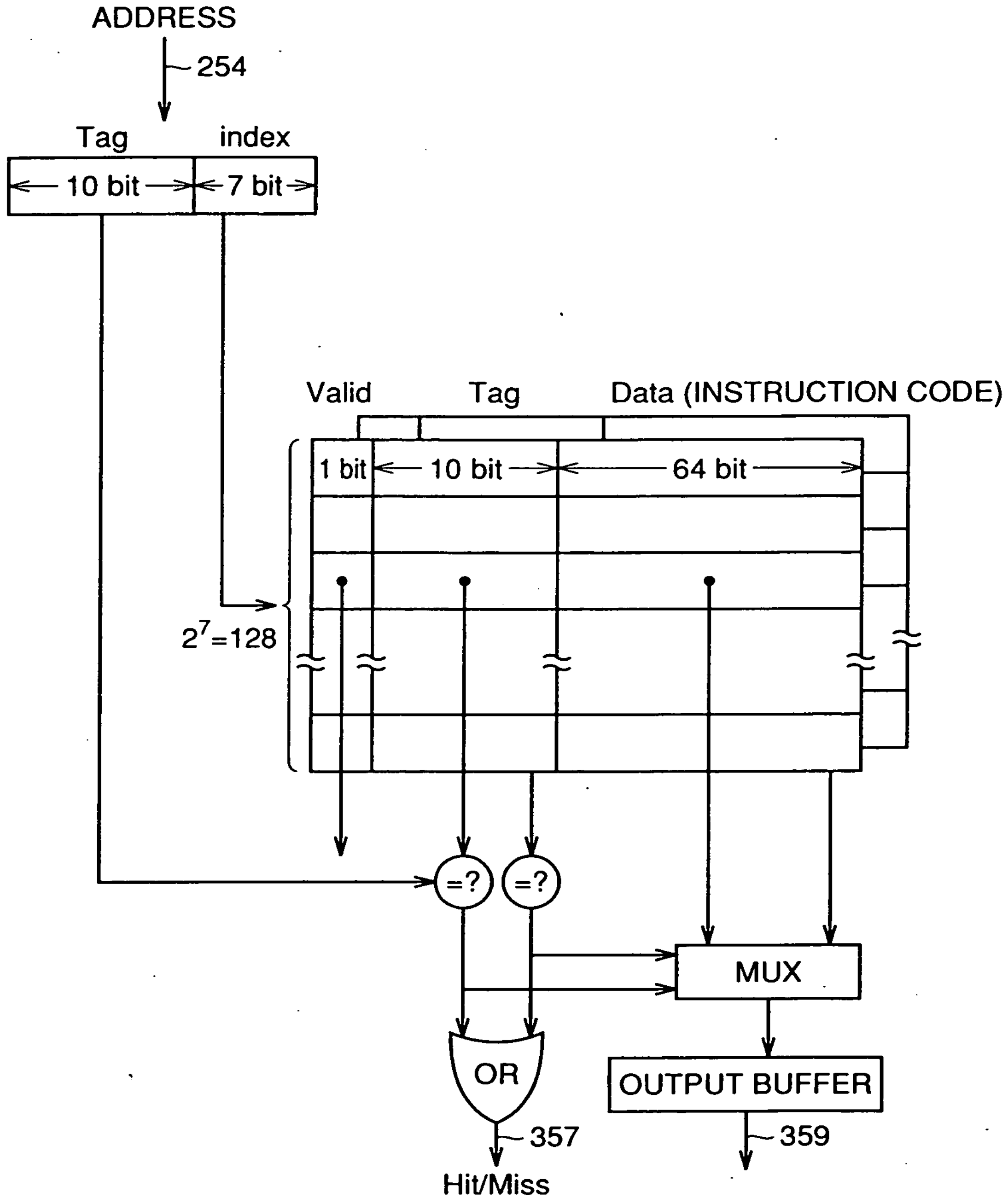

ADDRESS
254
Tag
index
10 bit
7 bit
Valid
Tag
Data (INSTRUCTION CODE)
1 bit
10 bit
64 bit
2^7=128
=?
=?
MUX
OR
OUTPUT BUFFER
357
359
Hit/Miss

FIG.15

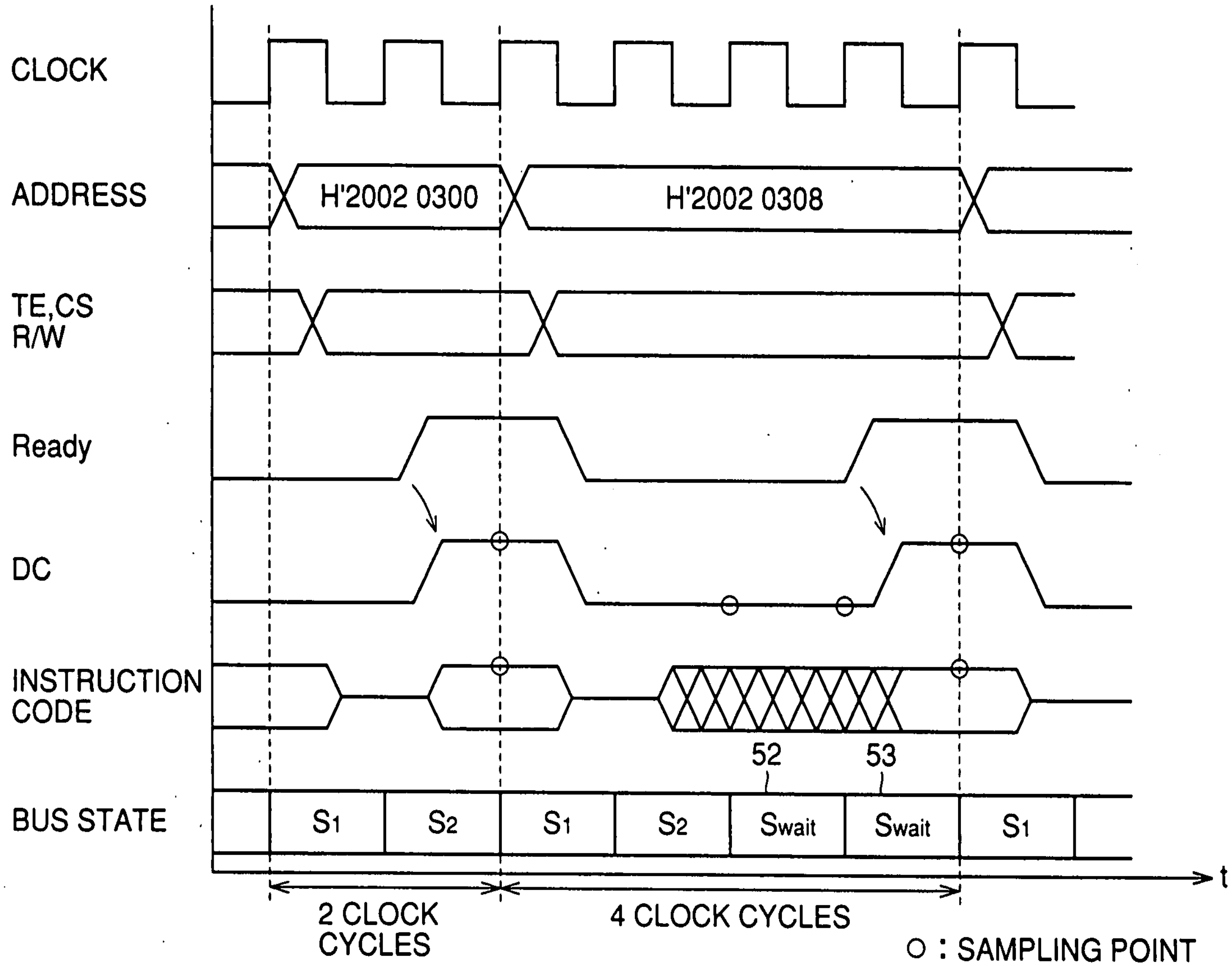
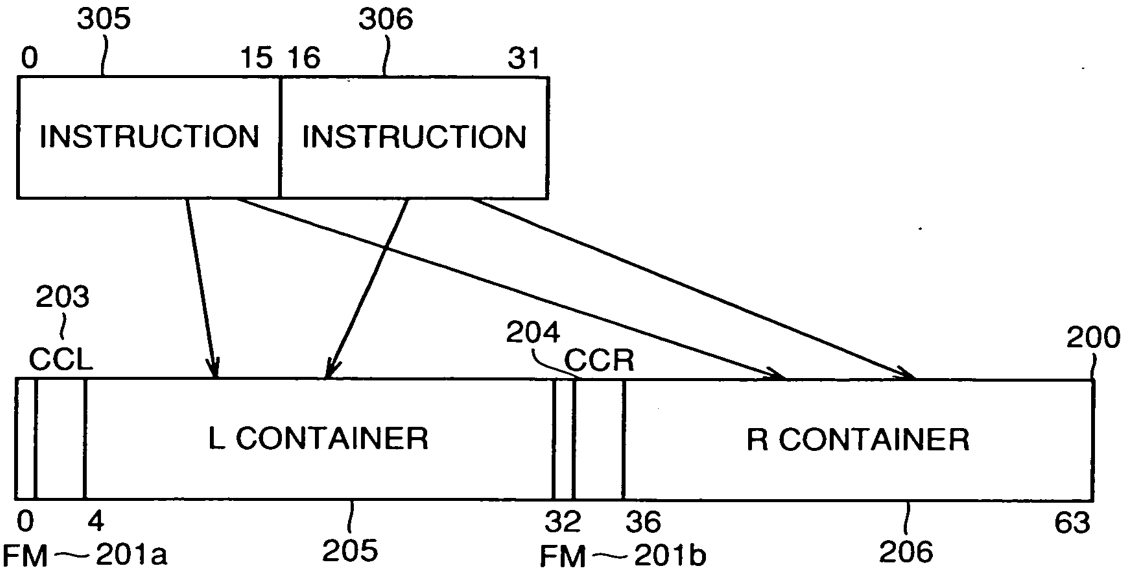

CLOCK
ADDRESS
H'2002 0300
H'2002 0308
TE,CS
R/W
Ready
DC
INSTRUCTION
CODE
52
53
BUS STATE
S1
S2
S1
S2
Swait
Swait
S1
t
2 CLOCK
CYCLES
4 CLOCK CYCLES
○ : SAMPLING POINT

FIG. 16



CCL=000	OTHER THAN BRAT, BRAF	CCR=000	OTHER THAN BRAT, BRAF
CCL=001	BRAT	CCR=001	BRAT
CCL=010	BRAF	CCR=010	BRAF

FM=00 EXECUTABLE IN PARALLEL WITH NO DEPENDENCE BETWEEN INSTRUCTIONS 305 AND 306

FM=01 NOT EXECUTABLE IN PARALLEL WITH DEPENDENCE BETWEEN INSTRUCTIONS 305 AND 306 OR RESTRICTION BY OPERATION UNIT (INSTRUCTION 305 TO L CONTAINER, INSTRUCTION 306 TO R CONTAINER)

FM=10 NOT EXECUTABLE IN PARALLEL WITH DEPENDENCE BETWEEN INSTRUCTIONS 305 AND 306 OR RESTRICTION BY OPERATION UNIT (INSTRUCTION 306 TO R CONTAINER, INSTRUCTION 305 TO L CONTAINER)

FIG.17

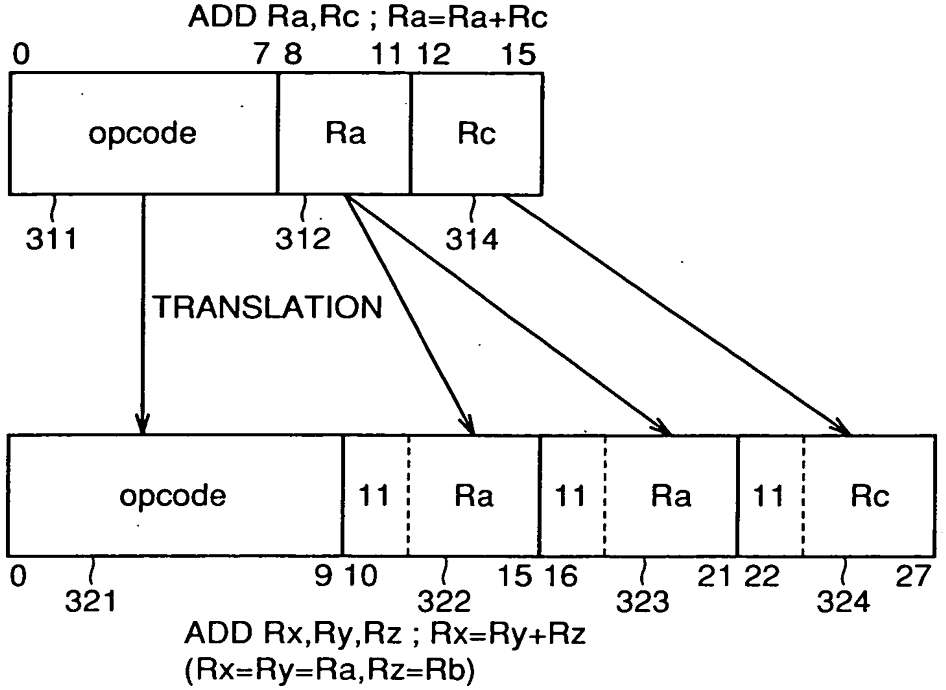

ADD Ra,Rc ; Ra=Ra+Rc
0
7 8
11 12
15
opcode
Ra
Rc
311
312
314
TRANSLATION
opcode
11
Ra
11
Ra
11
Rc
0
9 10
15 16
21 22
27
321
322
323
324
ADD Rx,Ry,Rz ; Rx=Ry+Rz
(Rx=Ry=Ra,Rz=Rb)

FIG.18

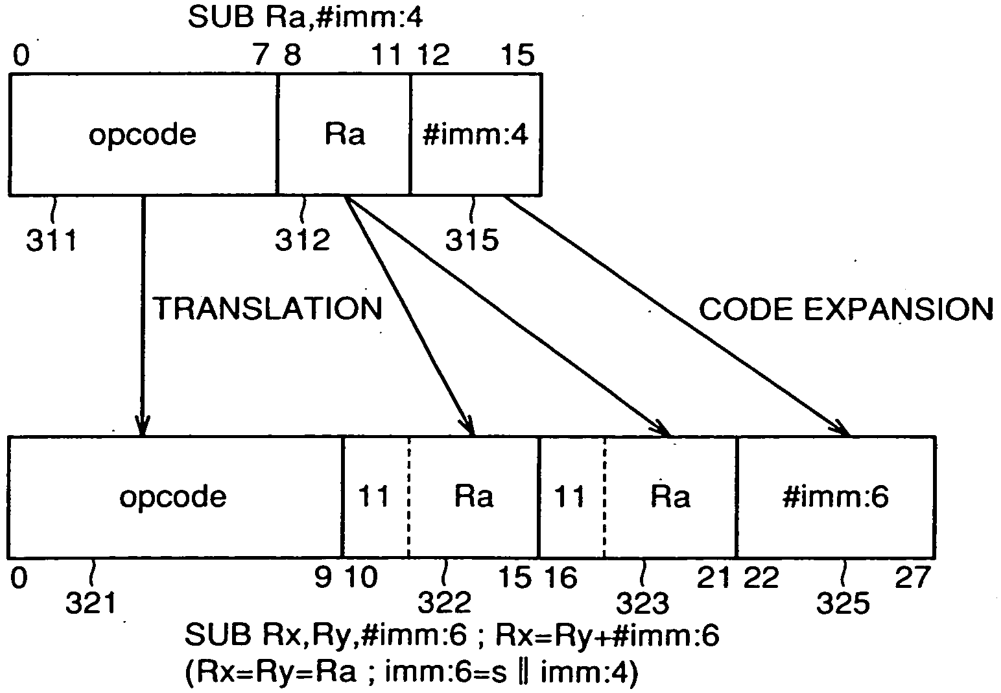

SUB Ra,#imm:4
0
7 8
11 12
15
opcode
Ra
#imm:4
311
312
315
TRANSLATION
CODE EXPANSION
opcode
11
Ra
11
Ra
#imm:6
0
9 10
15 16
21 22
27
321
322
323
325
SUB Rx,Ry,#imm:6 ; Rx=Ry+#imm:6
(Rx=Ry=Ra ; imm:6=s ‖ imm:4)

FIG. 19

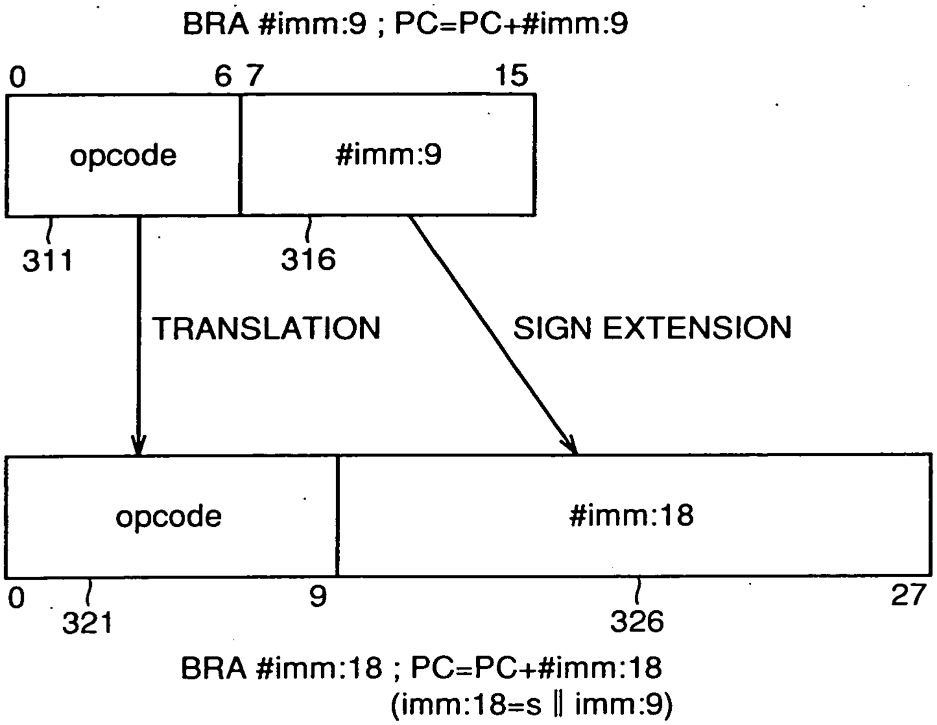
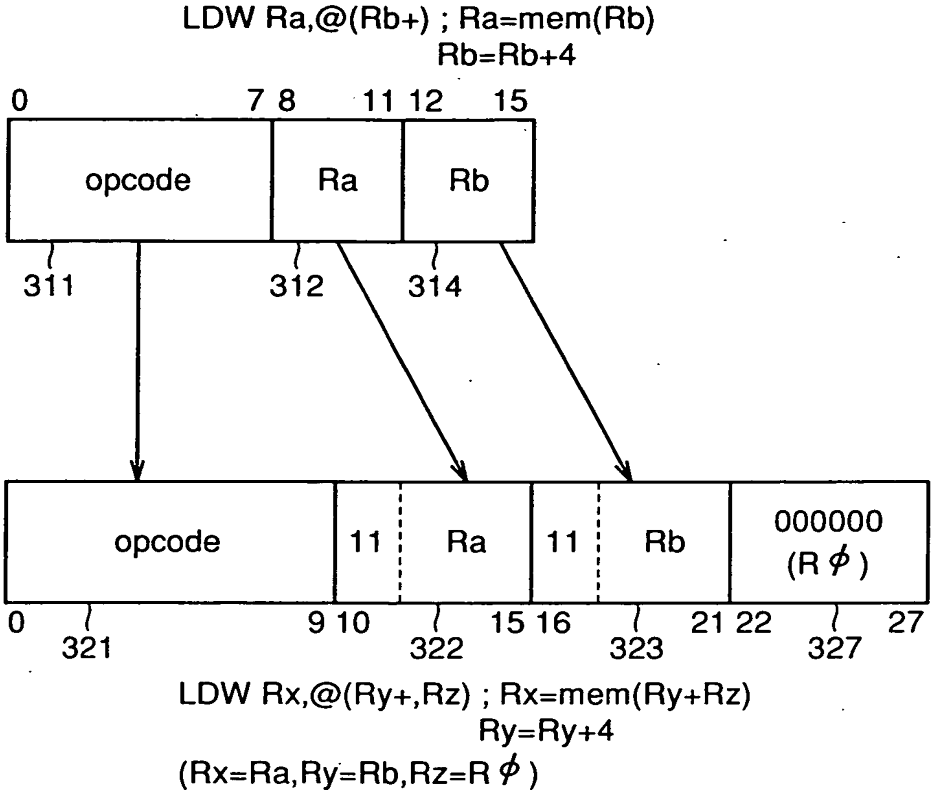
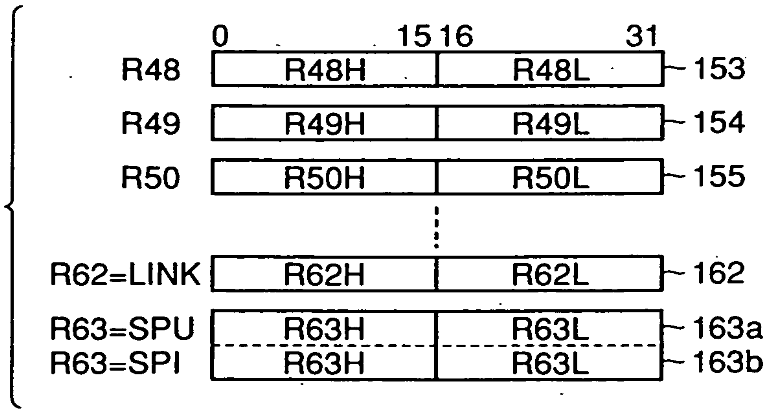

BRA #imm:9 ; PC=PC+#imm:9
0
6
7
15
opcode
#imm:9
311
316
TRANSLATION
SIGN EXTENSION
opcode
#imm:18
0
321
9
326
27
BRA #imm:18 ; PC=PC+#imm:18
(imm:18=s ∥ imm:9)

FIG.20

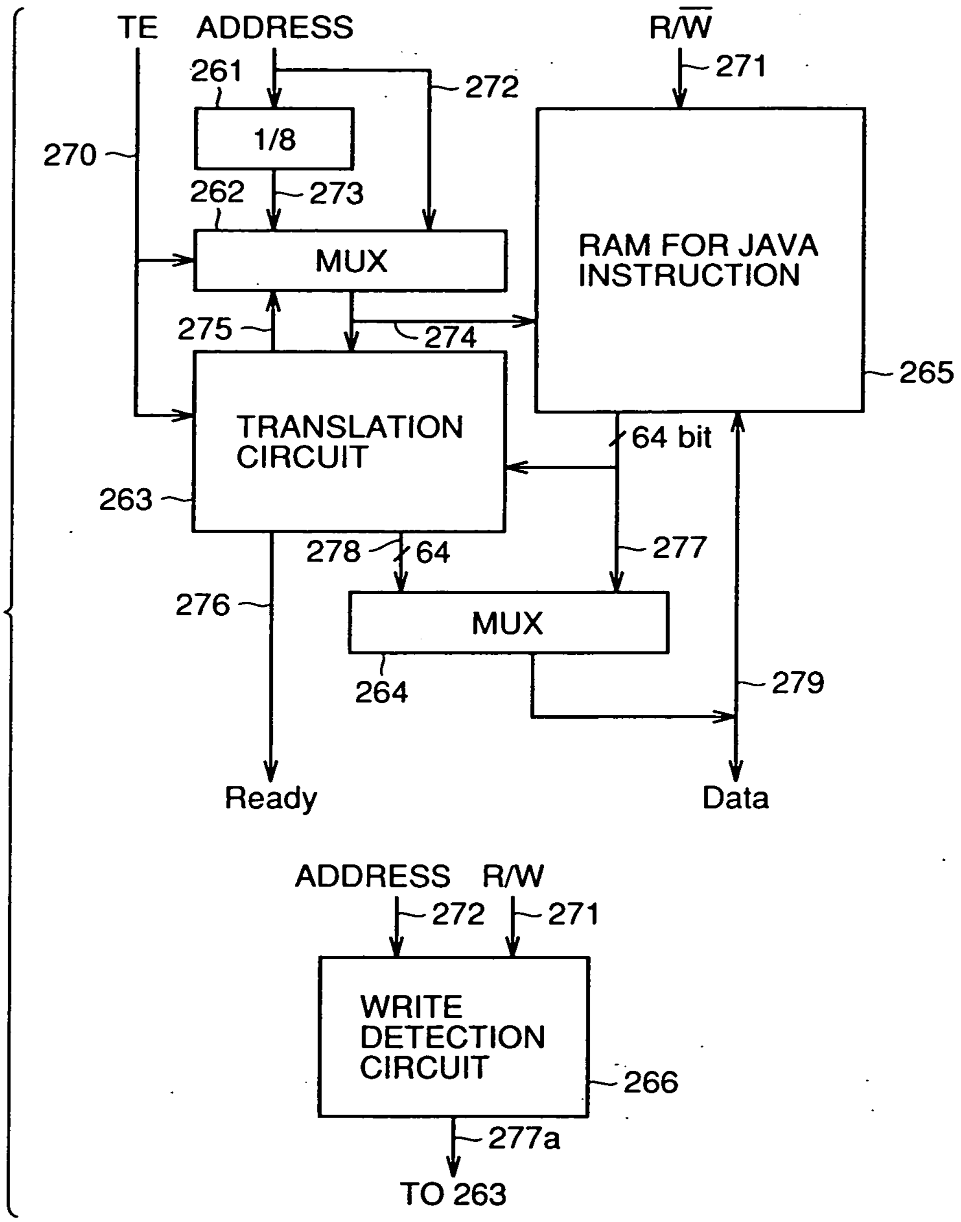


LDW Ra,@(Rb+) ; Ra=mem(Rb)
Rb=Rb+4
0
7 8
11 12
15
opcode
Ra
Rb
311
312
314
opcode
11
Ra
11
Rb
000000
(R ϕ)
0
9 10
15 16
21 22
27
321
322
323
327
LDW Rx,@(Ry+,Rz) ; Rx=mem(Ry+Rz)
Ry=Ry+4
(Rx=Ra,Ry=Rb,Rz=R ϕ)

FIG.21



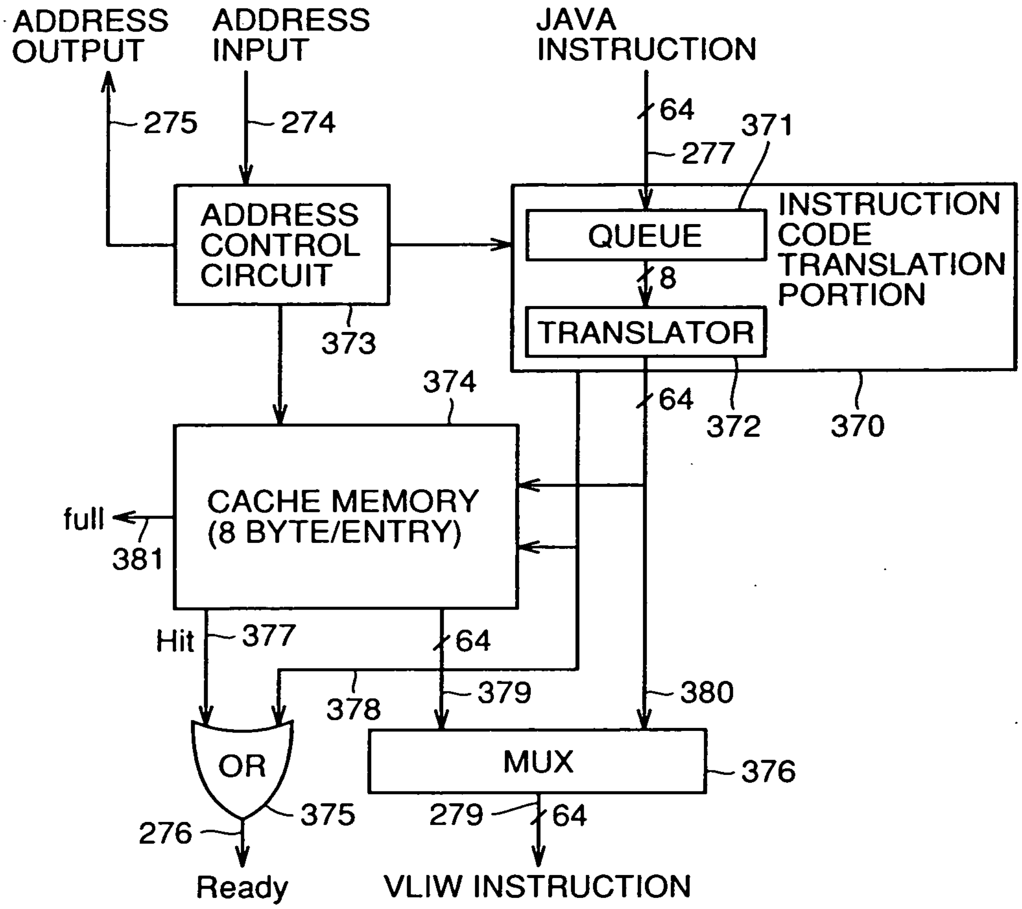
0
15 16
31
R48
R48H
R48L
153
R49
R49H
R49L
154
R50
R50H
R50L
155
R62=LINK
R62H
R62L
162
R63=SPU
R63H
R63L
163a
R63=SPI
R63H
R63L
163b



TE
ADDRESS
R/W̄
261
272
271
1/8
270
262
273
MUX
RAM FOR JAVA INSTRUCTION
275
274
265
TRANSLATION CIRCUIT
64 bit
263
278
64
277
276
MUX
264
279
Ready
Data
ADDRESS
R/W
272
271
WRITE DETECTION CIRCUIT
266
277a
TO 263

FIG.22

FIG.23



TRANSLATION CIRCUIT

FIG.24

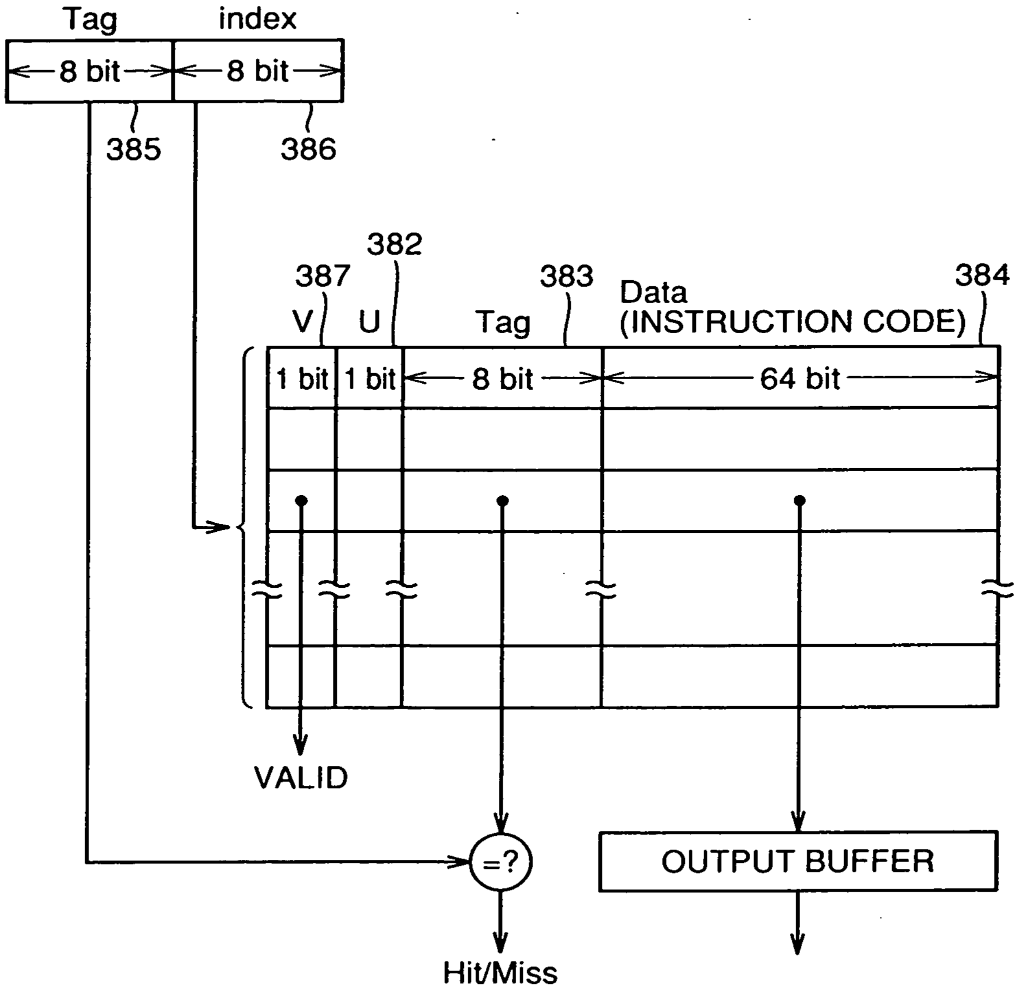

Tag
index
8 bit
8 bit
385
386
382
387
383
384
V
U
Tag
Data
(INSTRUCTION CODE)
1 bit
1 bit
8 bit
64 bit
VALID
=?
OUTPUT BUFFER
Hit/Miss

FIG.25

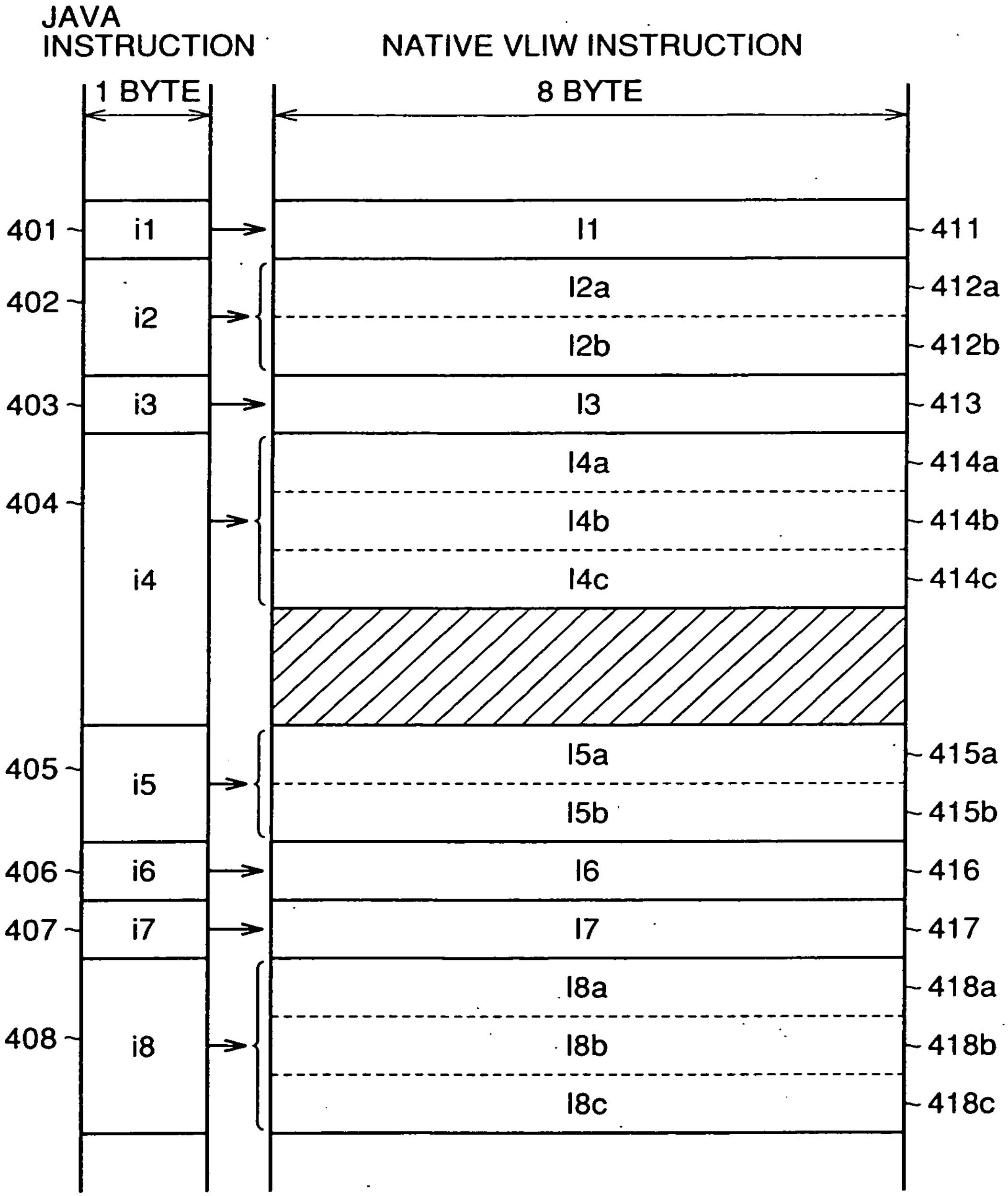
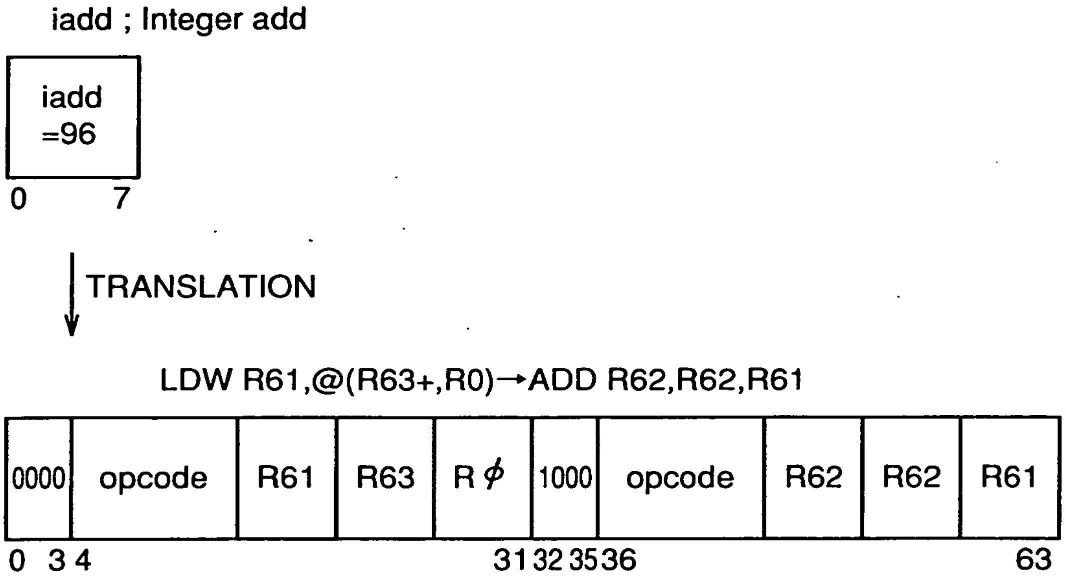
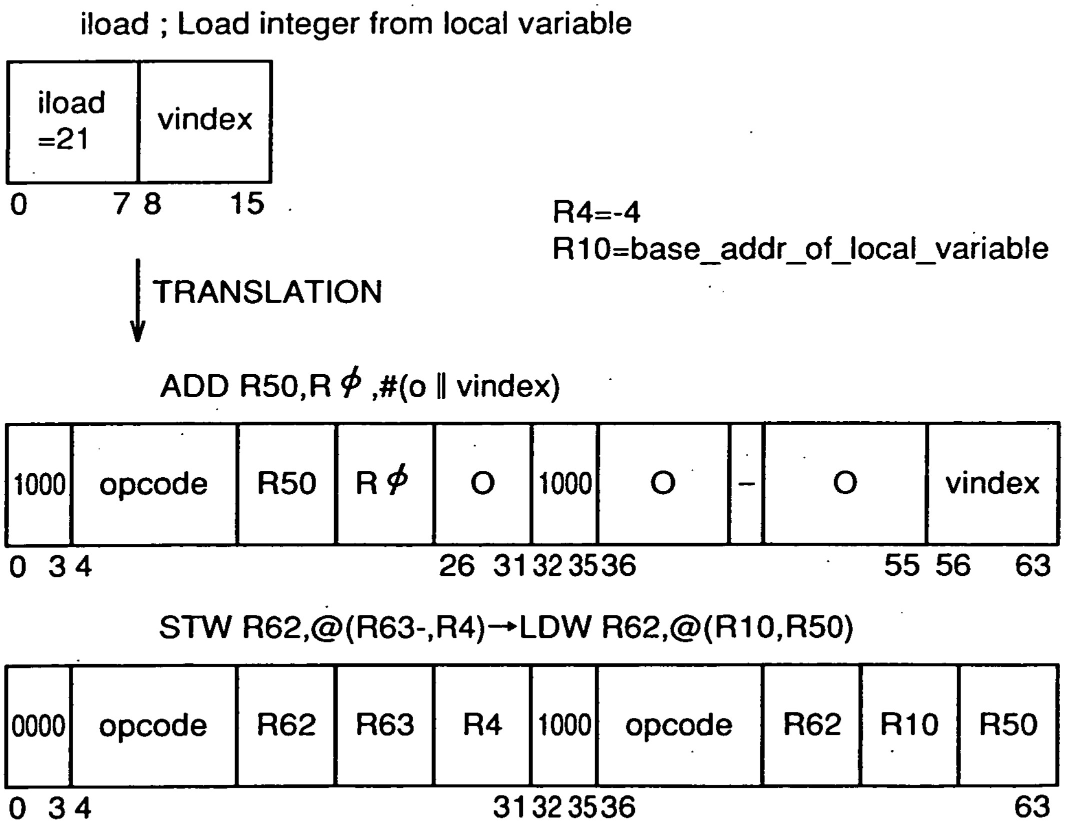

JAVA INSTRUCTION
NATIVE VLIW INSTRUCTION
1 BYTE
8 BYTE
401
i1
I1
411
402
i2
I2a
412a
I2b
412b
403
i3
I3
413
404
i4
I4a
414a
I4b
414b
I4c
414c
405
i5
I5a
415a
I5b
415b
406
i6
I6
416
407
i7
I7
417
408
i8
I8a
418a
I8b
418b
I8c
418c

FIG.26



EXAMPLE OF 1BYTE → 1 VLIW INSTRUCTION

FIG.27



EXAMPLE OF 2 BYTE → 2 VLIW INSTRUCTION

FIG.28

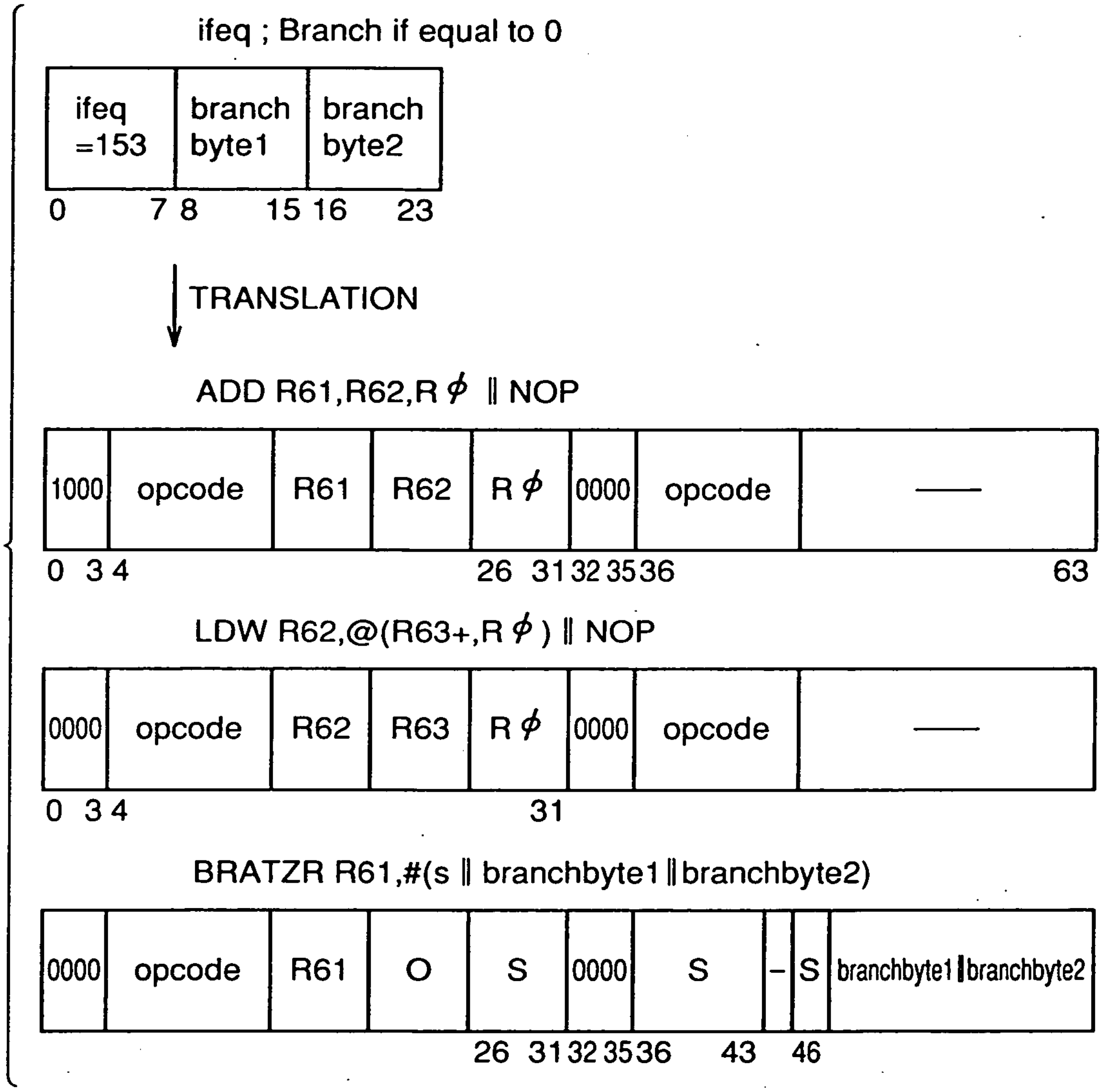

ifeq ; Branch if equal to 0
ifeq =153
branch byte1
branch byte2
0
7 8
15 16
23
TRANSLATION
ADD R61,R62,Rϕ ‖ NOP
1000
opcode
R61
R62
Rϕ
0000
opcode
——
0
3 4
26
31 32 35 36
63
LDW R62,@(R63+,Rϕ) ‖ NOP
0000
opcode
R62
R63
Rϕ
0000
opcode
——
0
3 4
31
BRATZR R61,#(s ‖ branchbyte1 ‖ branchbyte2)
0000
opcode
R61
O
S
0000
S
–
S
branchbyte1 ‖ branchbyte2
26
31 32 35 36
43
46

FIG.29

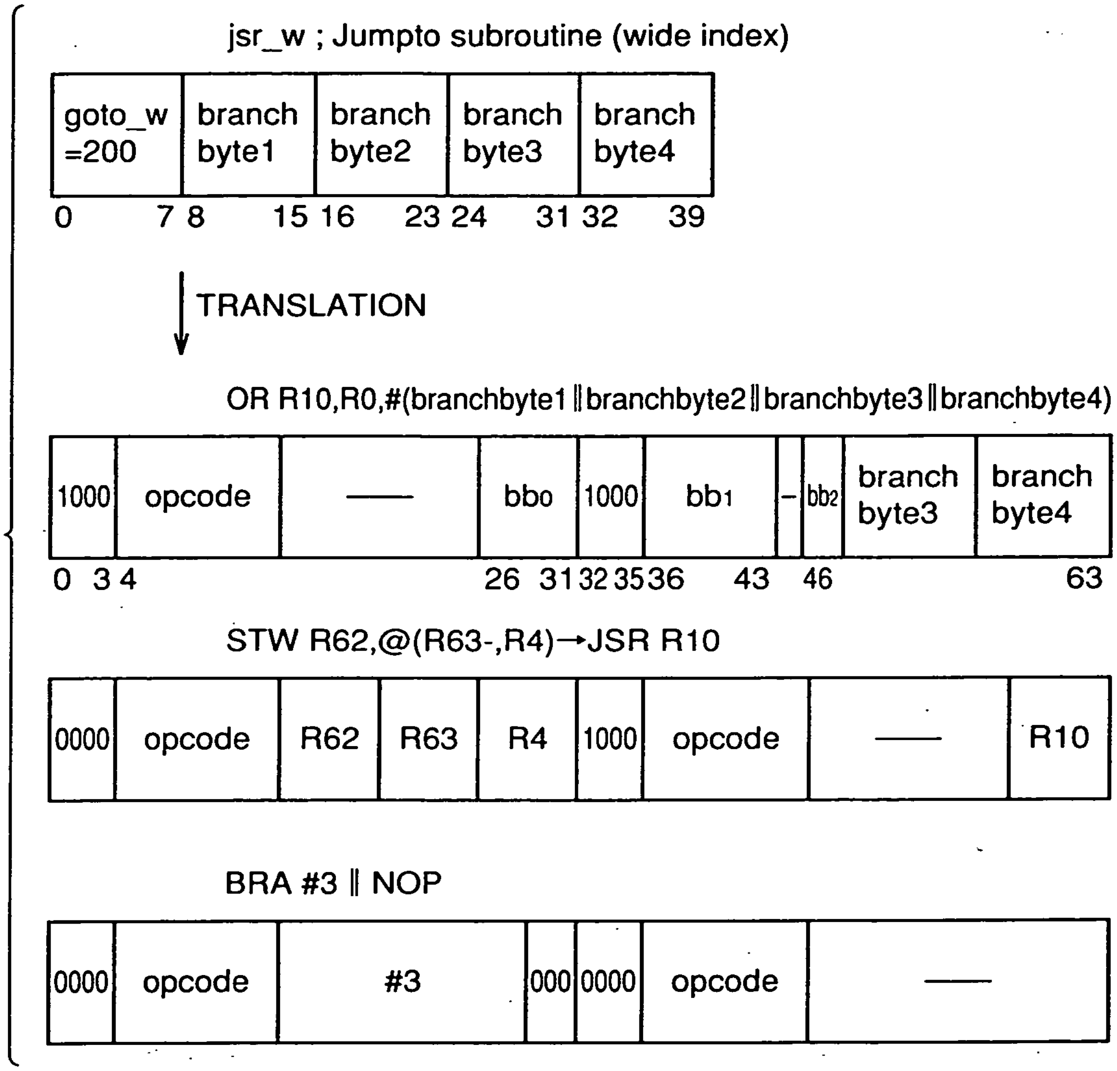

jsr_w ; Jumpto subroutine (wide index)
goto_w =200
branch byte1
branch byte2
branch byte3
branch byte4
0
7 8
15 16
23 24
31 32
39
TRANSLATION
OR R10,R0,#(branchbyte1 ‖ branchbyte2 ‖ branchbyte3 ‖ branchbyte4)
1000
opcode
—
bb0
1000
bb1
–
bb2
branch byte3
branch byte4
0 3 4
26 31 32 35 36
43
46
63
STW R62,@(R63-,R4)→JSR R10
0000
opcode
R62
R63
R4
1000
opcode
—
R10
BRA #3 ‖ NOP
0000
opcode
#3
000
0000
opcode
—

FIG.30

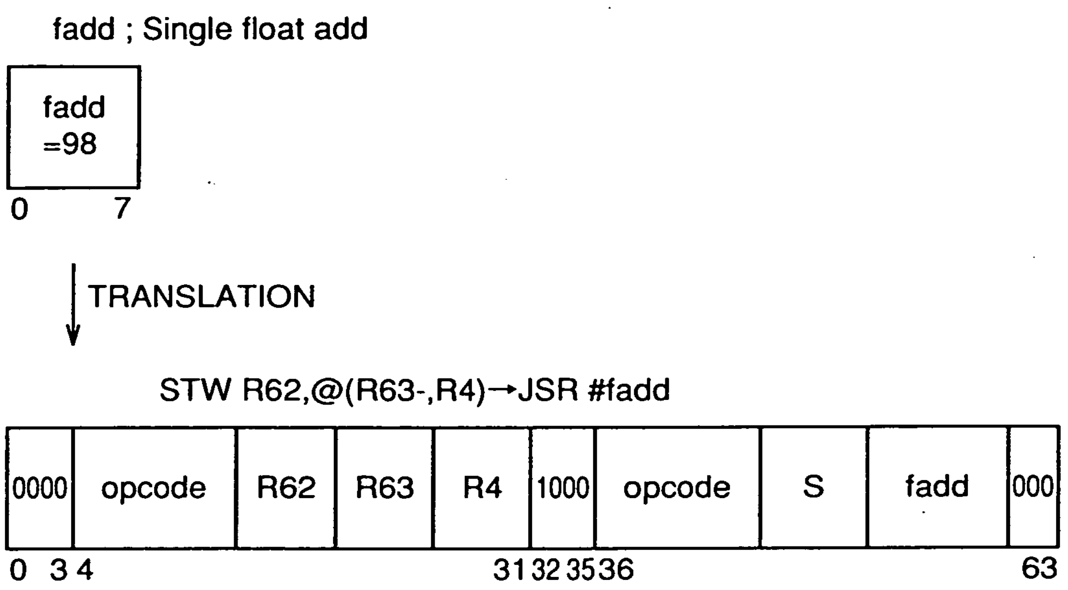

fadd ; Single float add
fadd
=98
0
7
TRANSLATION
STW R62,@(R63-,R4)→JSR #fadd
0000
opcode
R62
R63
R4
1000
opcode
S
fadd
000
0
3
4
31
32
35
36
63

FIG.31

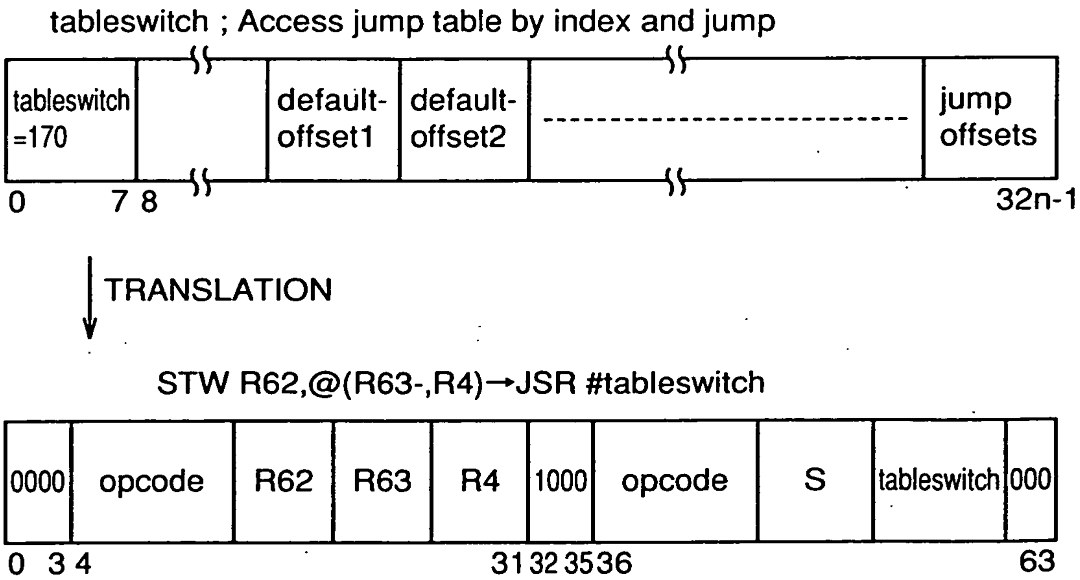
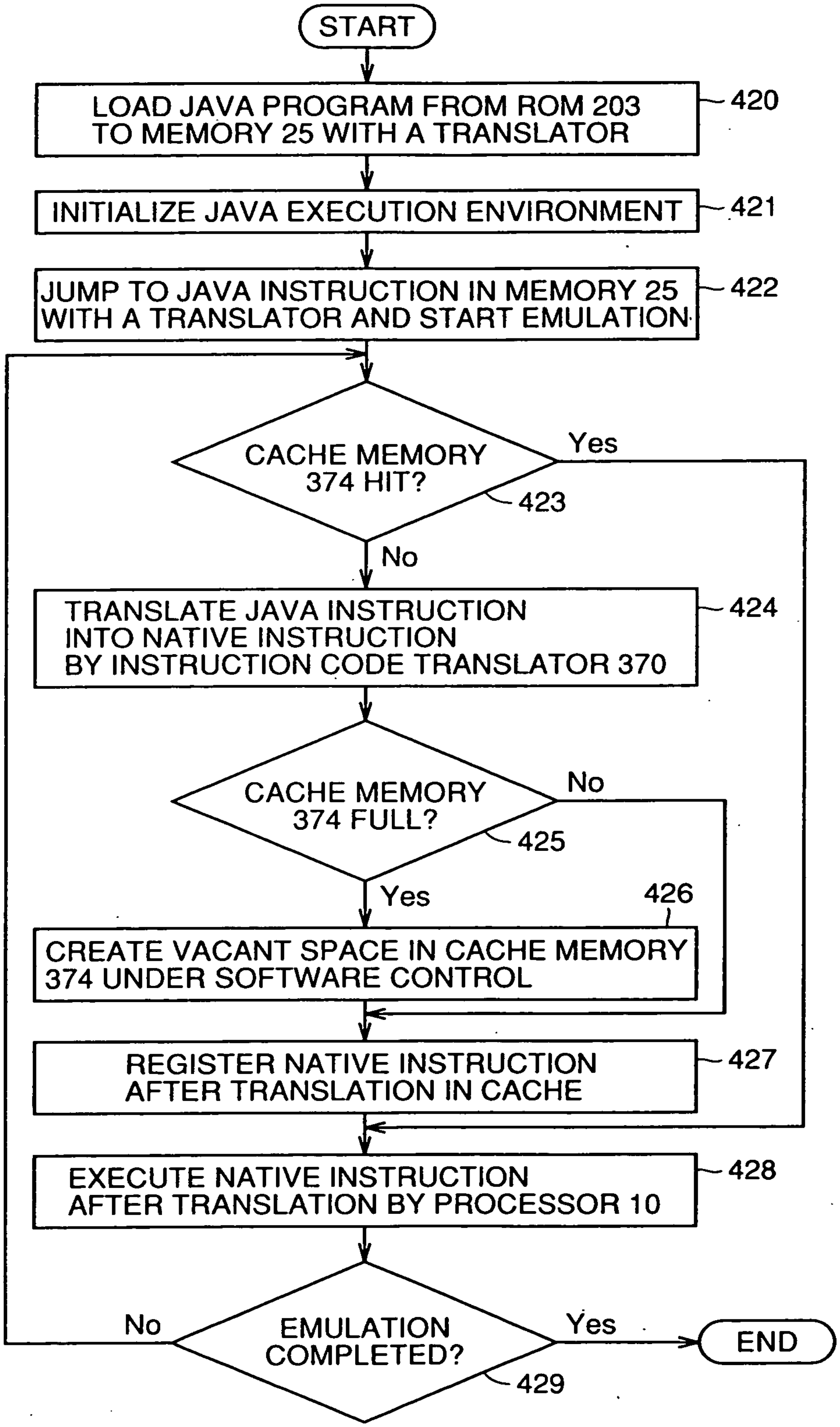

tableswitch ; Access jump table by index and jump
tableswitch
=170
default-
offset1
default-
offset2
jump
offsets
0
7
8
32n-1
TRANSLATION
STW R62,@(R63-,R4)→JSR #tableswitch
0000
opcode
R62
R63
R4
1000
opcode
S
tableswitch
000
0
3
4
31
32
35
36
63

FIG.32


START
LOAD JAVA PROGRAM FROM ROM 203 TO MEMORY 25 WITH A TRANSLATOR
420
INITIALIZE JAVA EXECUTION ENVIRONMENT
421
JUMP TO JAVA INSTRUCTION IN MEMORY 25 WITH A TRANSLATOR AND START EMULATION
422
CACHE MEMORY 374 HIT?
423
Yes
No
TRANSLATE JAVA INSTRUCTION INTO NATIVE INSTRUCTION BY INSTRUCTION CODE TRANSLATOR 370
424
CACHE MEMORY 374 FULL?
425
No
Yes
426
CREATE VACANT SPACE IN CACHE MEMORY 374 UNDER SOFTWARE CONTROL
REGISTER NATIVE INSTRUCTION AFTER TRANSLATION IN CACHE
427
EXECUTE NATIVE INSTRUCTION AFTER TRANSLATION BY PROCESSOR 10
428
No
EMULATION COMPLETED?
Yes
END
429